

FÍSICA E MOVIMENTO

Arnaldo Abrantes

Paulo Vieira

2019

REVISÕES SOBRE VECTORES

- Operações

- Soma: $\vec{w} = \vec{u} + \vec{v}$

- $\vec{w} = (8,6)$

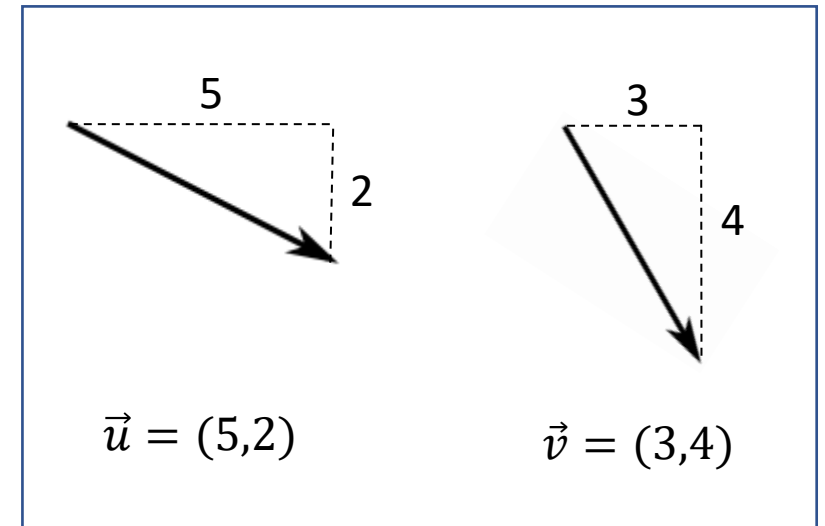
- Subtração: $\vec{w} = \vec{u} - \vec{v}$

- $\vec{w} = (2, -2)$

- Multiplicação por um escalar: $\vec{w} = \vec{u} \times n$

- $\vec{w} = (15,6)$ para $n = 3$

- Norma: $||\vec{v}|| = \sqrt{v_x^2 + v_y^2}$

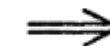
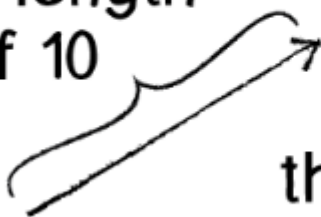


REVISÕES SOBRE VECTORES

- Operações
 - Normalização – processo a partir do qual é possível obter um vector unitário, ou seja, com norma 1. Não é modificada a direcção neste processo


$$\hat{u} = \frac{\vec{u}}{||\vec{u}||}$$

this vector
has a length
of 10



the process of
normalization

this vector has
a length of 1



VECTORES - PROCESSING

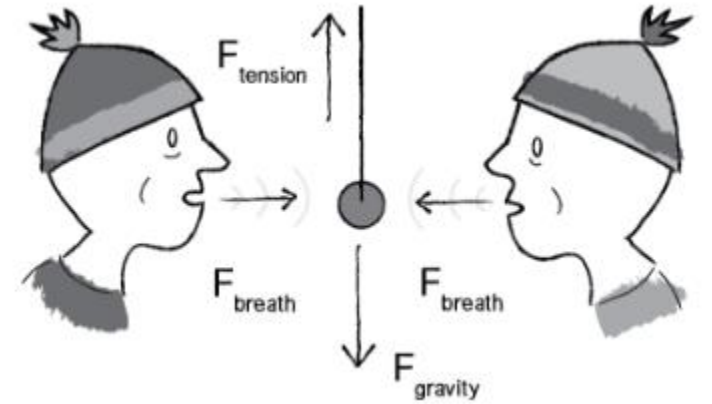
- `add()` — add vectors
- `sub()` — subtract vectors
- `mult()` — scale the vector with multiplication
- `div()` — scale the vector with division
- `mag()` — calculate the magnitude of a vector
- `setMag()` - set the magnitude of a vector
- `normalize()` — normalize the vector to a unit length of 1
- `limit()` — limit the magnitude of a vector
- `heading()` — the 2D heading of a vector expressed as an angle
- `rotate()` — rotate a 2D vector by an angle
- `lerp()` — linear interpolate to another vector
- `dist()` — the Euclidean distance between two vectors (considered as points)
- `angleBetween()` — find the angle between two vectors
- `dot()` — the dot product of two vectors
- `cross()` — the cross product of two vectors (only relevant in three dimensions)
- `random2D()` - make a random 2D vector
- `random3D()` - make a random 3D vector

PVector
<code>+x : float</code> <code>+y : float</code> <code>+z : float</code> <code>#array : float[]" [0..*]</code>
<code><<constructor>>+PVector()</code> <code>+random2D() : PVector</code> <code>+random3D() : PVector</code> <code>+fromAngle(angle : float) : PVector</code> <code>+fromAngle(angle : float, target : PVector) : PVector</code> <code>+mag() : float</code> <code>+magSq() : float</code> <code>+add(v : PVector) : PVector</code> <code>+sub(v : PVector) : PVector</code> <code>+mult(n : float) : PVector</code> <code>+div(n : float) : PVector</code> <code>+dist(v : PVector) : float</code> <code>+dot(v : PVector) : float</code> <code>+cross(v : PVector) : PVector</code> <code>+normalize() : PVector</code> <code>+limit(max : float) : PVector</code> <code><<setter>>+setMag(len : float) : PVector</code> <code>+heading() : float</code> <code><<JavaElement>>+heading2D() : float{JavaAnnotations = "@Deprecated"}</code> <code>+rotate(theta : float) : PVector</code> <code>+angleBetween(v1 : PVector, v2 : PVector) : float</code> <code><<JavaElement>>+toString() : String{JavaAnnotations = "@Override"}</code> <code>+array() : float[]"</code> <code><<JavaElement>>+equals(obj : Object) : boolean{JavaAnnotations = "@Override"}</code> <code><<JavaElement>>+hashCode() : int{JavaAnnotations = "@Override"}</code> <code>...</code>

LEIS DE NEWTON

- Primeira lei de Newton

“Um objeto em repouso permanece em repouso e um objeto em movimento permanece em movimento com velocidade e direção constantes a não ser que, sobre ele, atue outra força”



LEIS DE NEWTON

- Terceira lei de Newton

“Para cada ação existe uma reação de igual intensidade e sentido oposto”

LEIS DE NEWTON

- Segunda lei de Newton

“Uma força f a atuar num objeto de massa m dá-lhe uma aceleração a de acordo com a equação $\vec{F} = m \cdot \vec{a}$ ”

```
void applyForce(PVector force) {  
    force.div(mass);  
    acceleration.add(force);  
}
```

Newton's second law (with force accumulation and mass)

QUESTÕES TEMÁTICAS

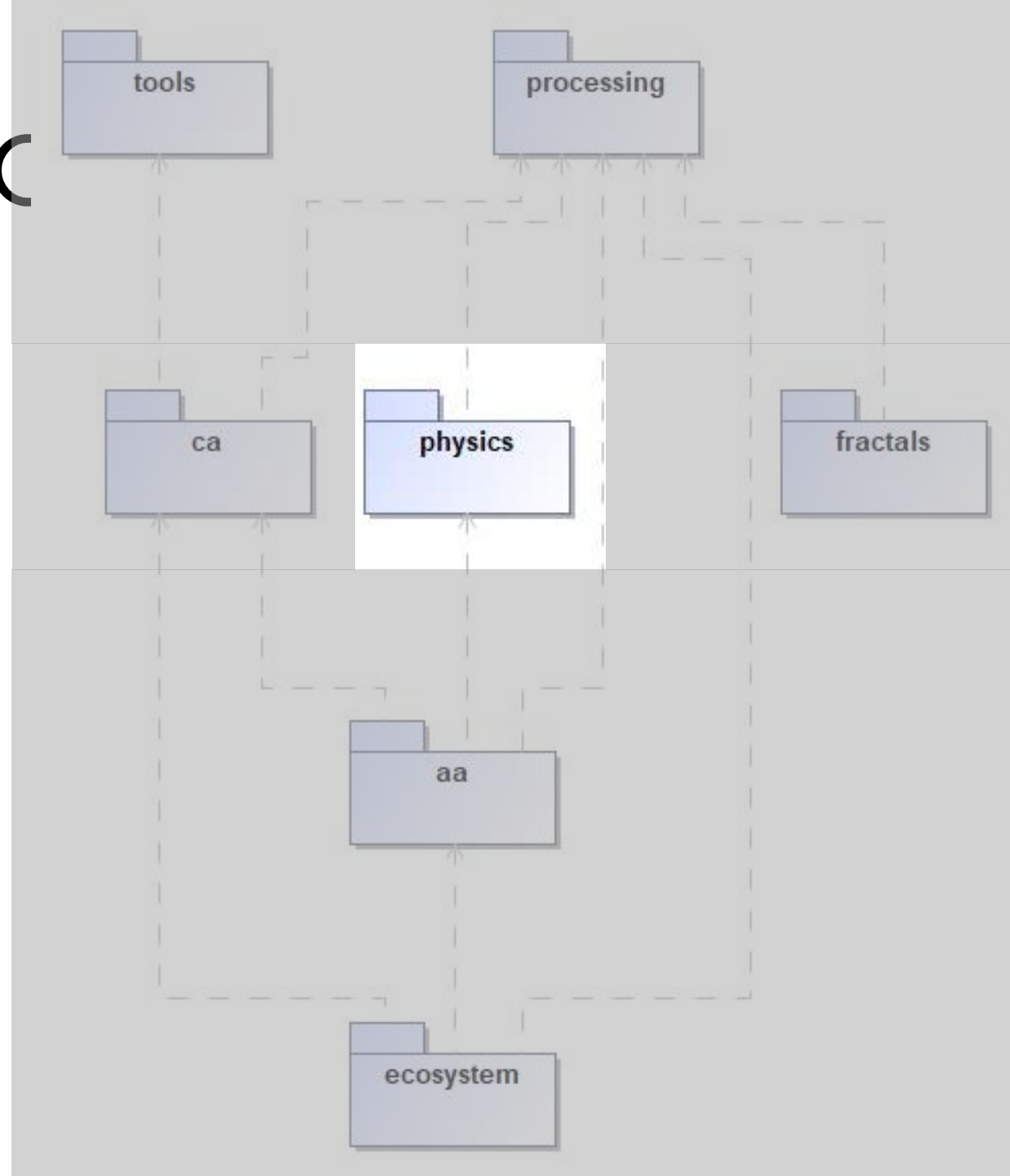
- Como simular uma bola a cair?
- Será que a bola demora o mesmo tempo a atravessar o ar e a água?
- Qual é a velocidade de impacto no solo de um paraquedista que salte de 2000m? Quanto tempo demorará a atingir o solo?
- Como simular o movimento dos planetas em torno do Sol?

ESTRUTURA DE APLICAÇÃO JAVA PARA SIMULAÇÃO

- Qual a estrutura necessária para suportar estas simulações em JAVA/Processing?
 - Objeto
 - Comportamento no Ar
 - Comportamento na Água

EXERCÍCIOS PRÁTICOS

- Package *physics*

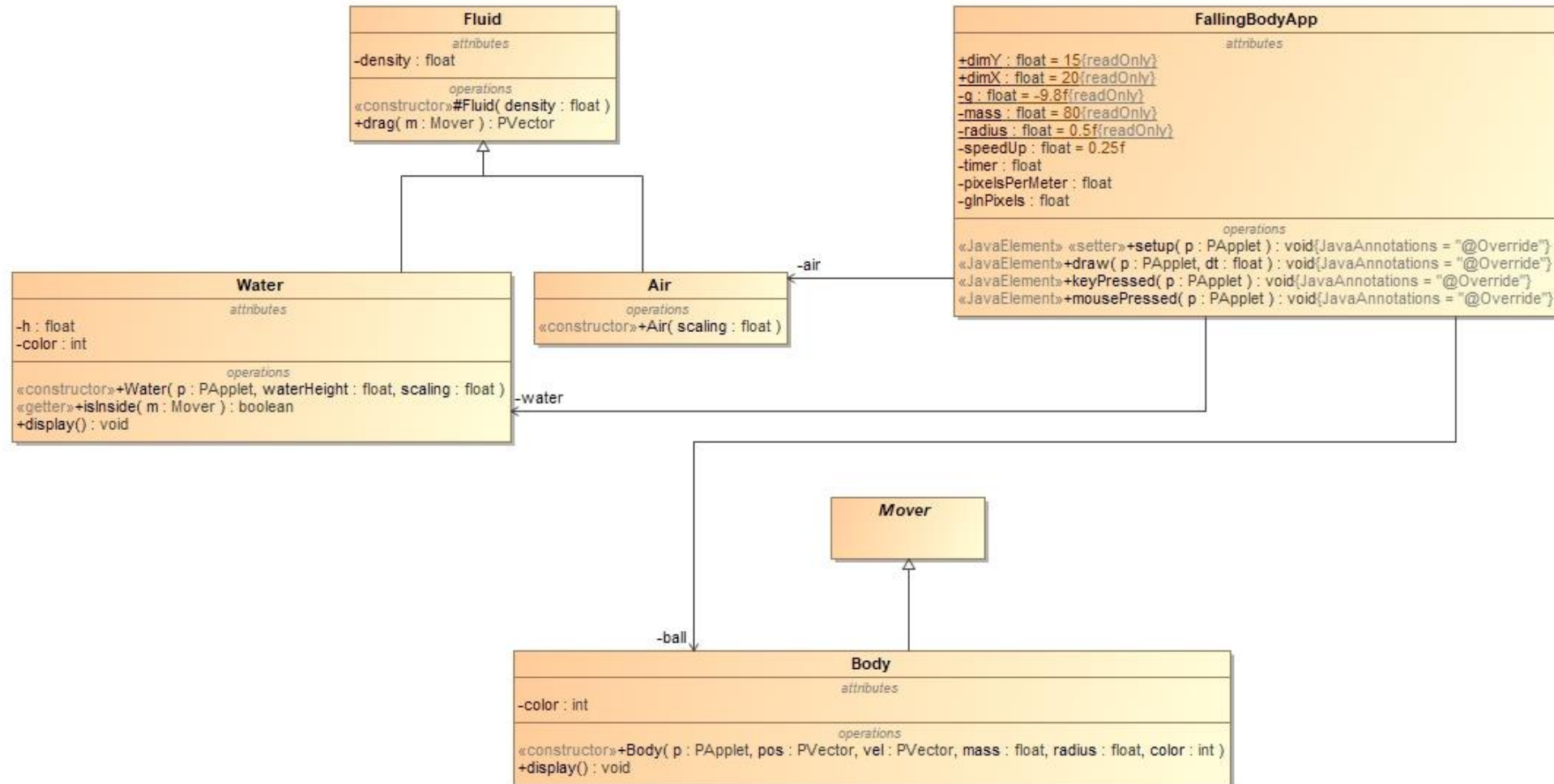


CLASSE MOVER – “A BOLA”

- *ApplyForce*
 - Como aplicar uma força? Ver segunda lei de Newton
- *Move*
 - Como movimentar o objeto?
 - Mudar a sua posição... mas diretamente nas coordenadas?

Mover	
attributes	
#pos : PVector	
#vel : PVector	
#acc : PVector	
#mass : float	
#radius : float	
operations	
«constructor»#Mover(pos : PVector, vel : PVector, mass : float, radius : float)	
+applyForce(f : PVector) : void	
+move(dt : float) : void	
«getter»+getPos() : PVector	
«getter»+getVel() : PVector	
«setter»+setVel(vel : PVector) : void	
«getter»+getMass() : float	
«getter»+getRadius() : float	

RESTANTES CLASSES

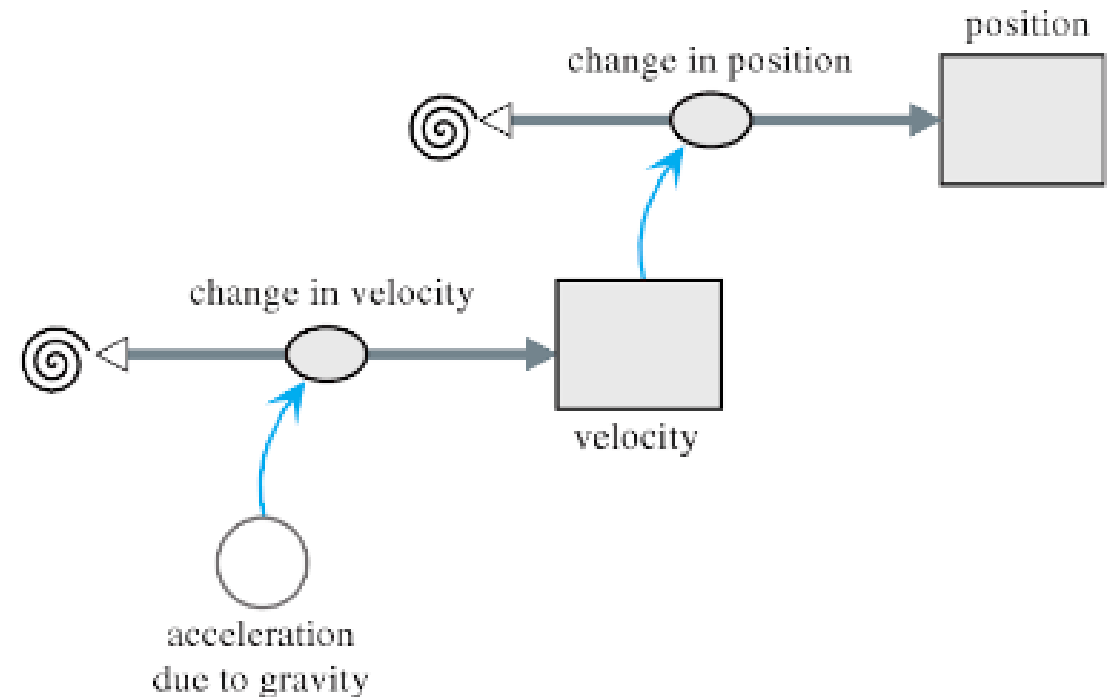


OBJETO SOB A ACÇÃO DA GRAVIDADE

- $v(t) = \frac{ds}{dt}$

- $a(t) = \frac{dv}{dt}$

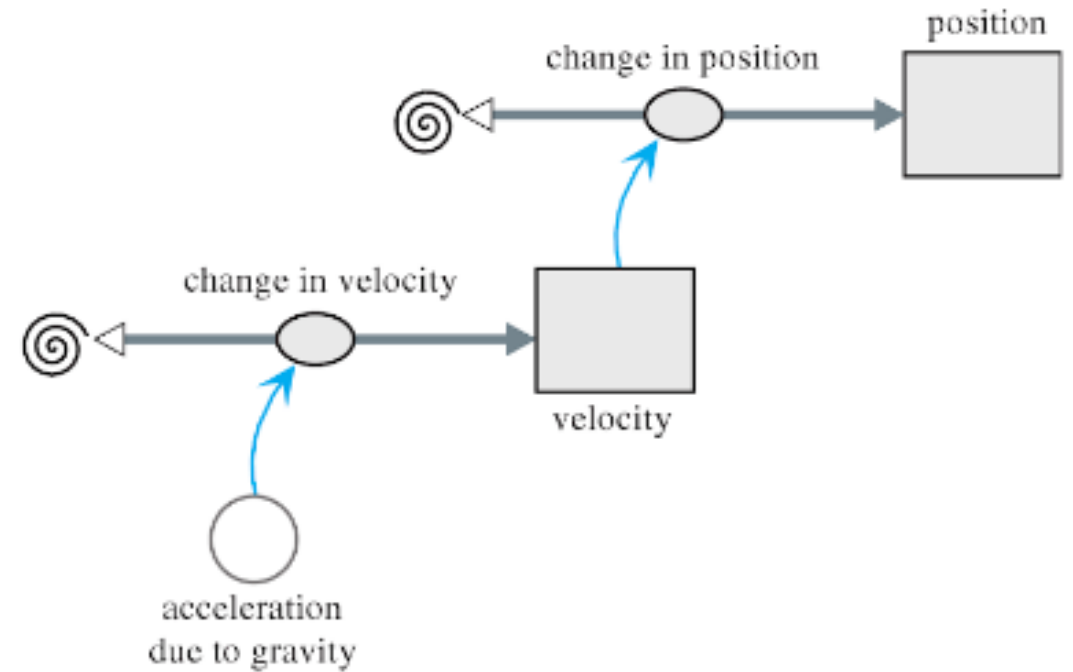
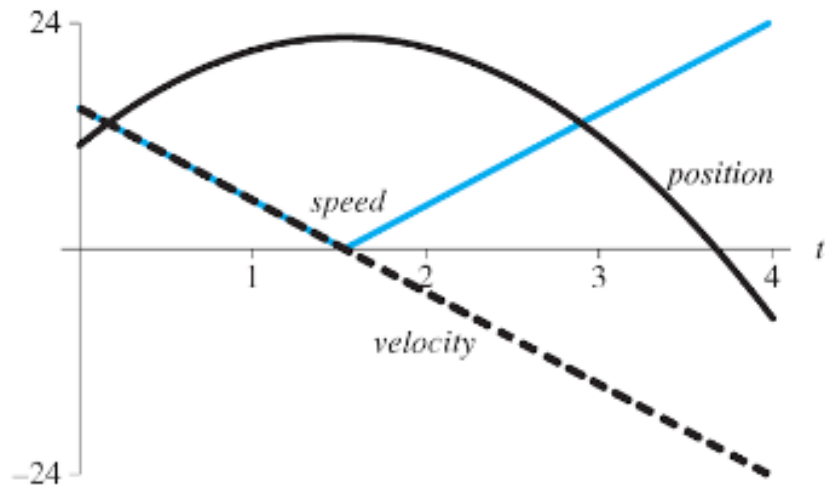
- Aceleração da gravidade:
 - $g = -9.81m/s^2$



OBJETO SOB A ACÇÃO DA GRAVIDADE

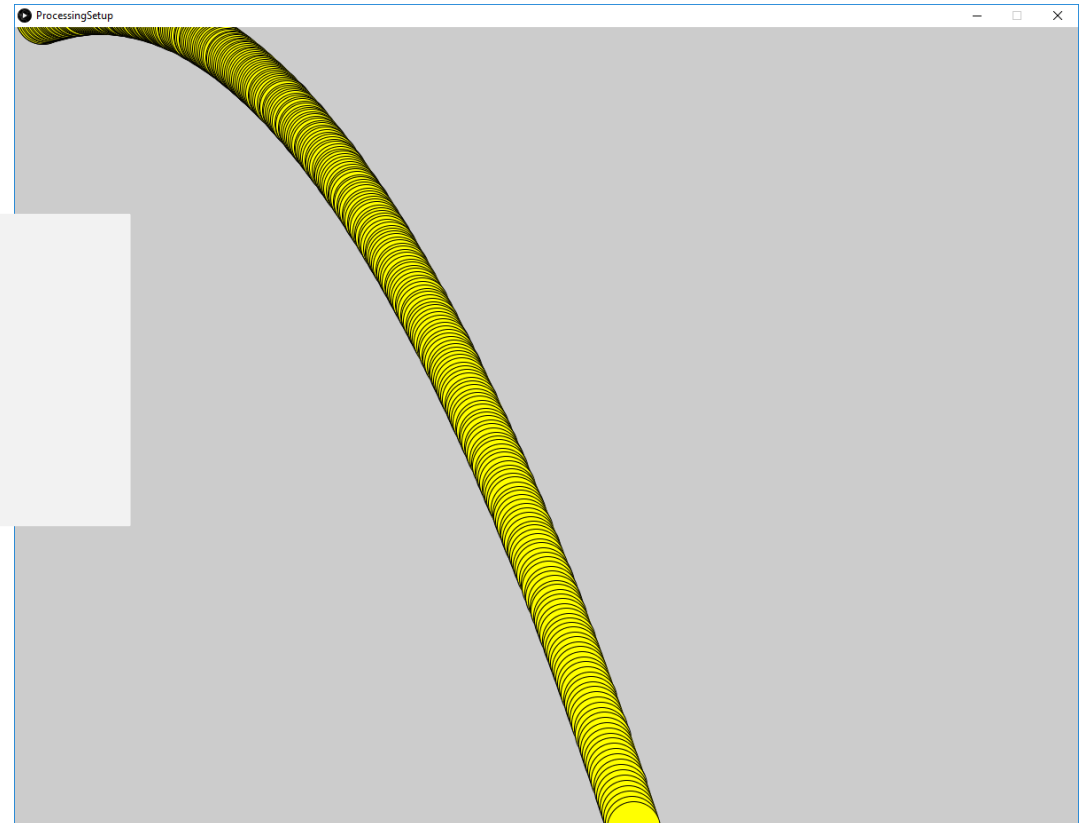
Exemplo:

- $v_0 = 15 \text{ m/s}$
- $p_0 = 11 \text{ m}$



OBJETO SOB A ACÇÃO DA GRAVIDADE

```
float pixelsPerMeter = p.height/dimY;  
float gInPixels = -g*pixelsPerMeter;  
  
PVector f = new PVector(0, mass*gInPixels);  
ball.applyForce(f);
```



OBJETO SOB A AÇÃO DA GRAVIDADE COM ARRASTO

- Arrasto é a força a que o objeto é sujeito quando atravessa determinado fluído
 - tende a atrasar o movimento
 - depende do meio (água, ar)

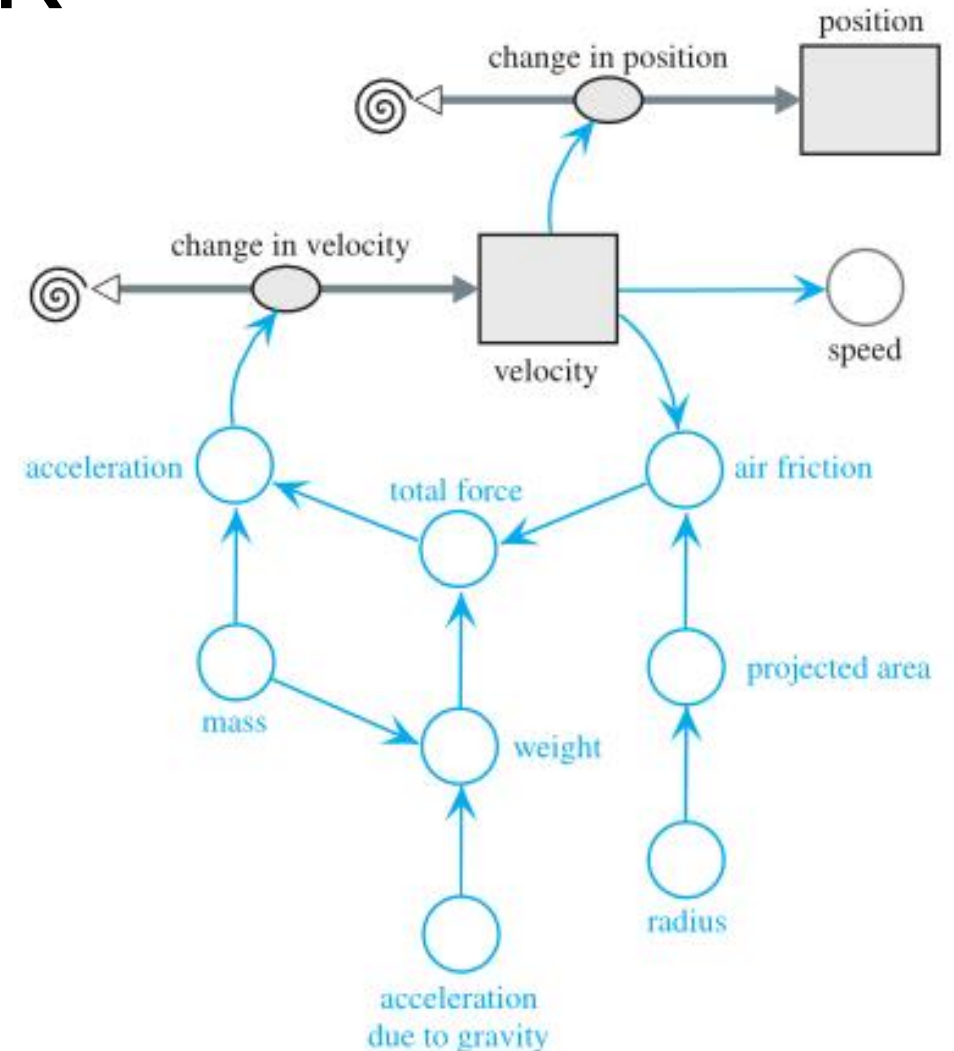
<i>Name</i>	<i>Formula</i>	<i>Meanings of Symbols</i>	<i>When to Use</i>
Stokes's friction	$F = kv$	k constant v velocity	Very small object moving slowly through fluid
Newtonian friction	$F = 0.5CDAv^2$	C coefficient of drag D density of fluid A object's projected area in direction of movement v velocity	Larger objects moving faster through fluid
Newtonian friction through air	$F = 0.65Av^2$	A object's projected area in direction of movement v velocity	Larger objects with $C = 1$ moving faster through sea-level air

OBJETO SOB A AÇÃO DA GRAVIDADE COM ARRASTO – NO AR

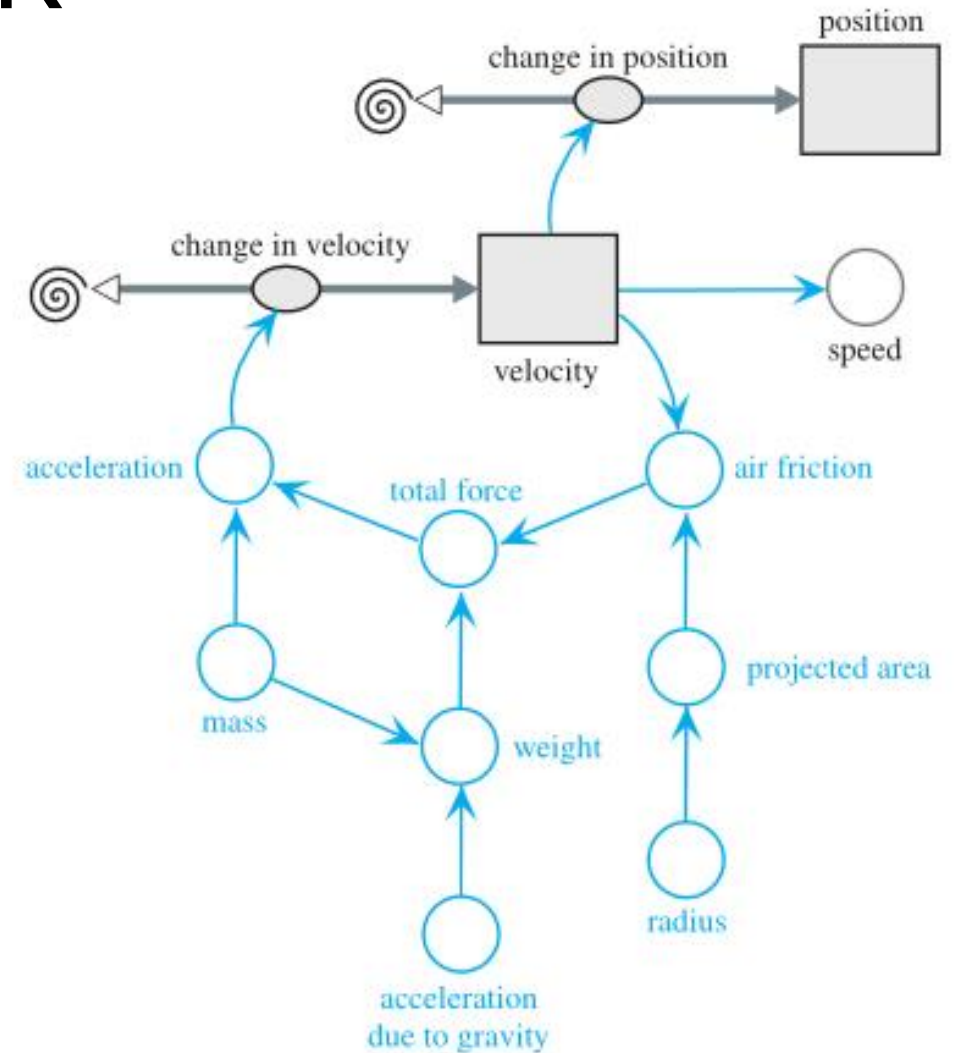
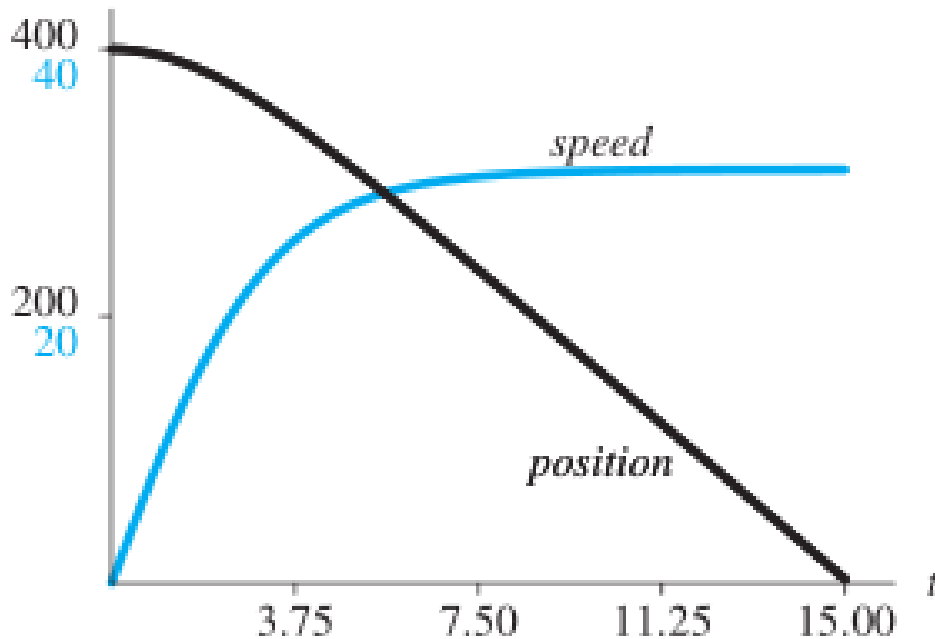
- Comportamento do objeto de acordo com a equação de Newton:

- $F = -0.65Av|v|$

mass = 0.5 kg
acceleration_due_to_gravity = -9.81 m/s²
radius = 0.05 m
weight = *mass* * *acceleration_due_to_gravity*
projected_area = 3.14159 * *radius*²
air_friction = -0.65 * *projected_area* * *velocity* * ABS(*velocity*)
total_force = *weight* + *air_friction*
acceleration = *total_force* / *mass*
change_in_velocity = *acceleration*
change_in_position = *velocity*
speed = ABS(*velocity*)
velocity(0) = 0 m/s
velocity(*t*) = *velocity*(*t* - Δ*t*) + (*change_in_velocity*) * Δ*t*
position(0) = 400 m
position(*t*) = *position*(*t* - Δ*t*) + (*change_in_position*) * Δ*t*

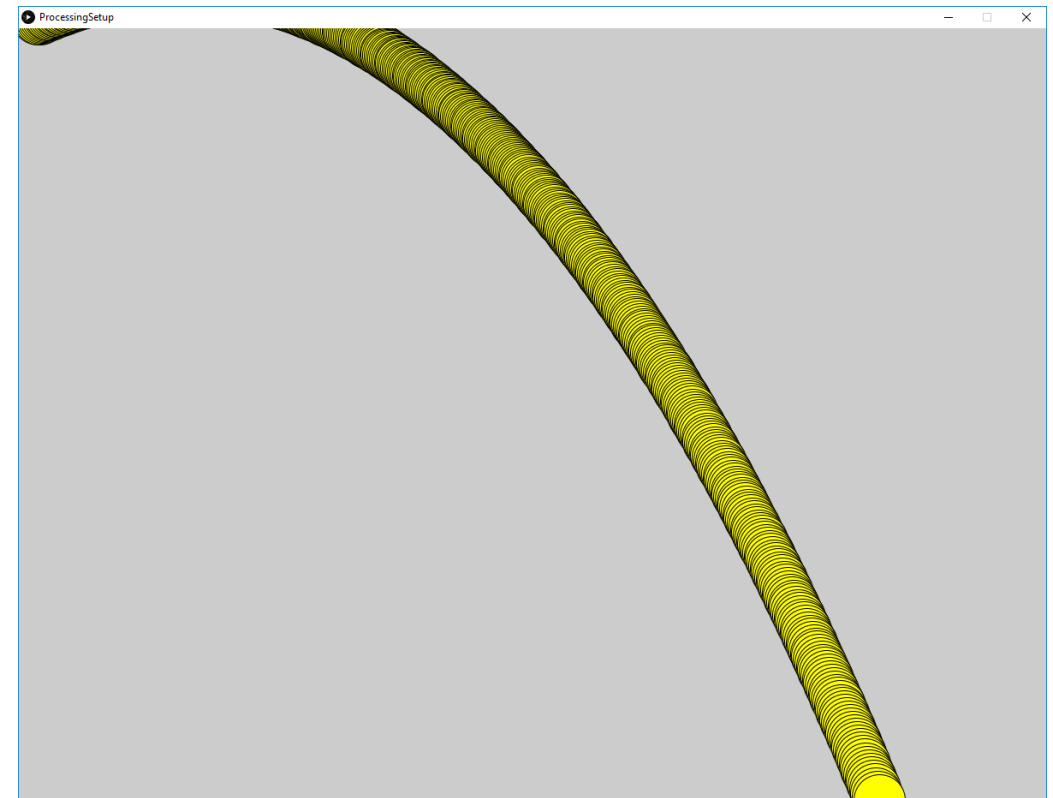


OBJETO SOB A AÇÃO DA GRAVIDADE COM ARRASTO – NO AR



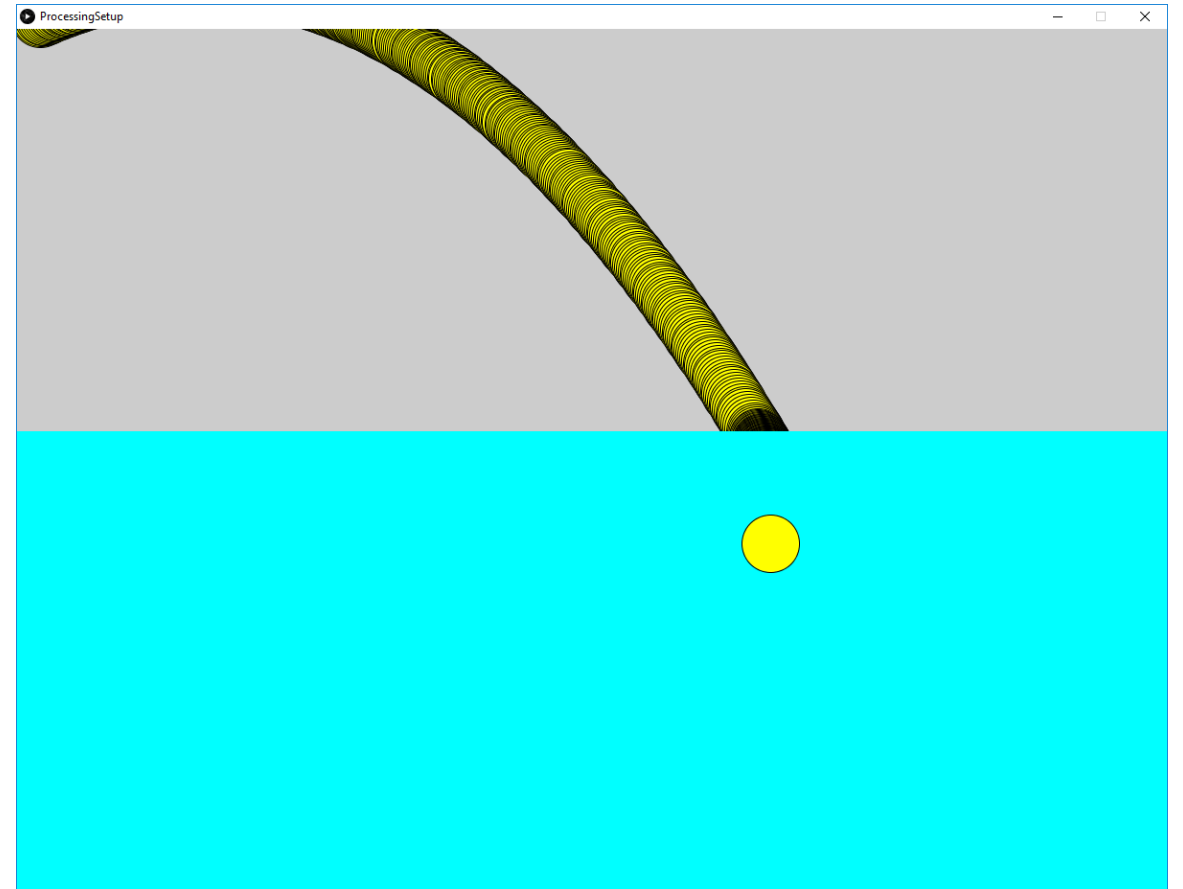
OBJETO SOB A ACÇÃO DA GRAVIDADE COM ARRASTO – NO AR

```
float pixelsPerMeter = p.height/dimY;  
float gInPixels = -g*pixelsPerMeter;  
  
PVector f = new PVector(0, mass*gInPixels);  
ball.applyForce(f);  
  
f = air.drag(ball);  
ball.applyForce(f);
```



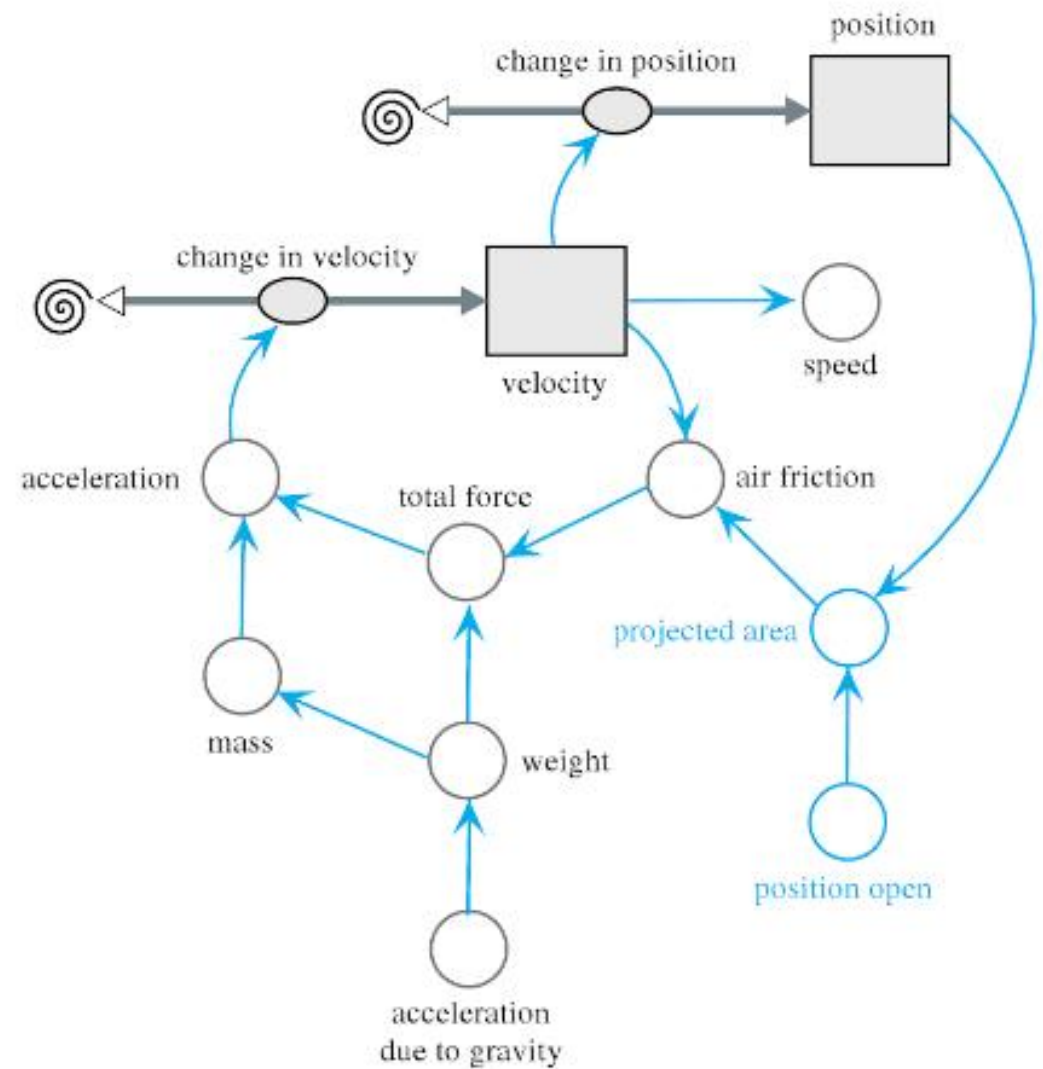
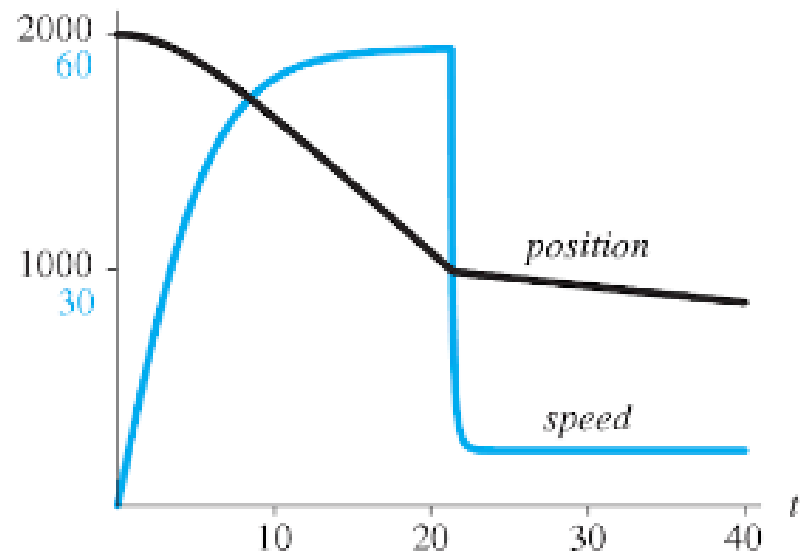
OBJETO SOB A ACÇÃO DA GRAVIDADE COM ARRASTO – NA ÁGUA

```
float pixelsPerMeter = p.height/dimY;  
float gInPixels = -g*pixelsPerMeter;  
  
PVector f = new PVector(0, mass*gInPixels);  
ball.applyForce(f);  
  
if (water.isInside(ball))  
    f = water.drag(ball);  
else  
    f = air.drag(ball);  
ball.applyForce(f);
```



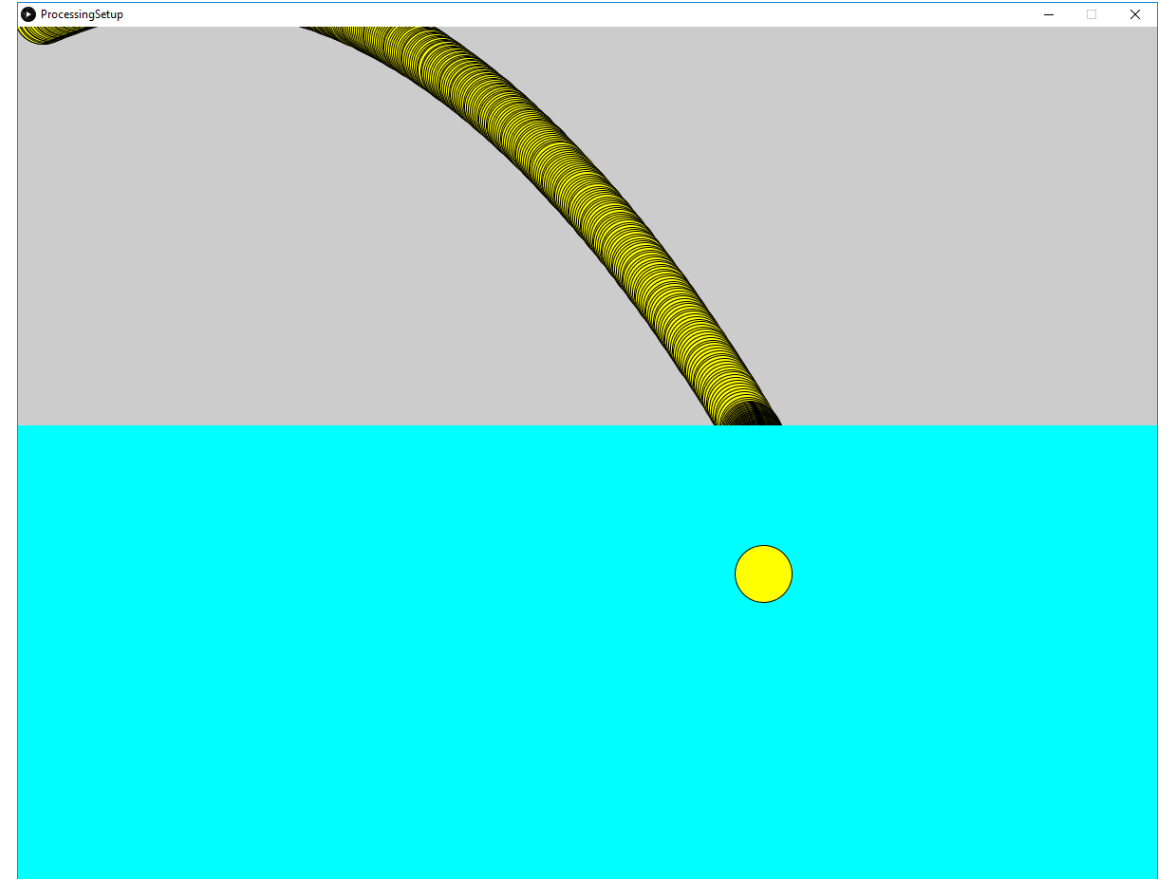
PARAQUEDISTA

```
if (position > position_open)
  projected_area ← 0.4
else
  projected_area ← 28
```



EXERCÍCIOS

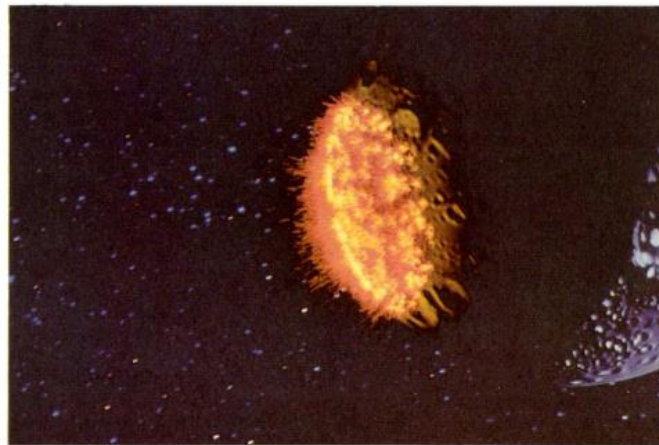
- Crie os modelos anteriores utilizando a ferramenta de modelação Anylogic
- Implemente, utilizando a linguagem JAVA e Processing, a aplicação *FallingBodyApp*



SISTEMA DE PARTÍCULAS

"A particle system is a collection of many many minute particles that together represent a fuzzy object. Over a period of time, particles are generated into a system, move and change from within the system, and die from the system."

– William Reeves, "Particle Systems—A Technique for Modeling a Class of Fuzzy Objects," ACM Transactions on Graphics 2:2 (April 1983), 92.

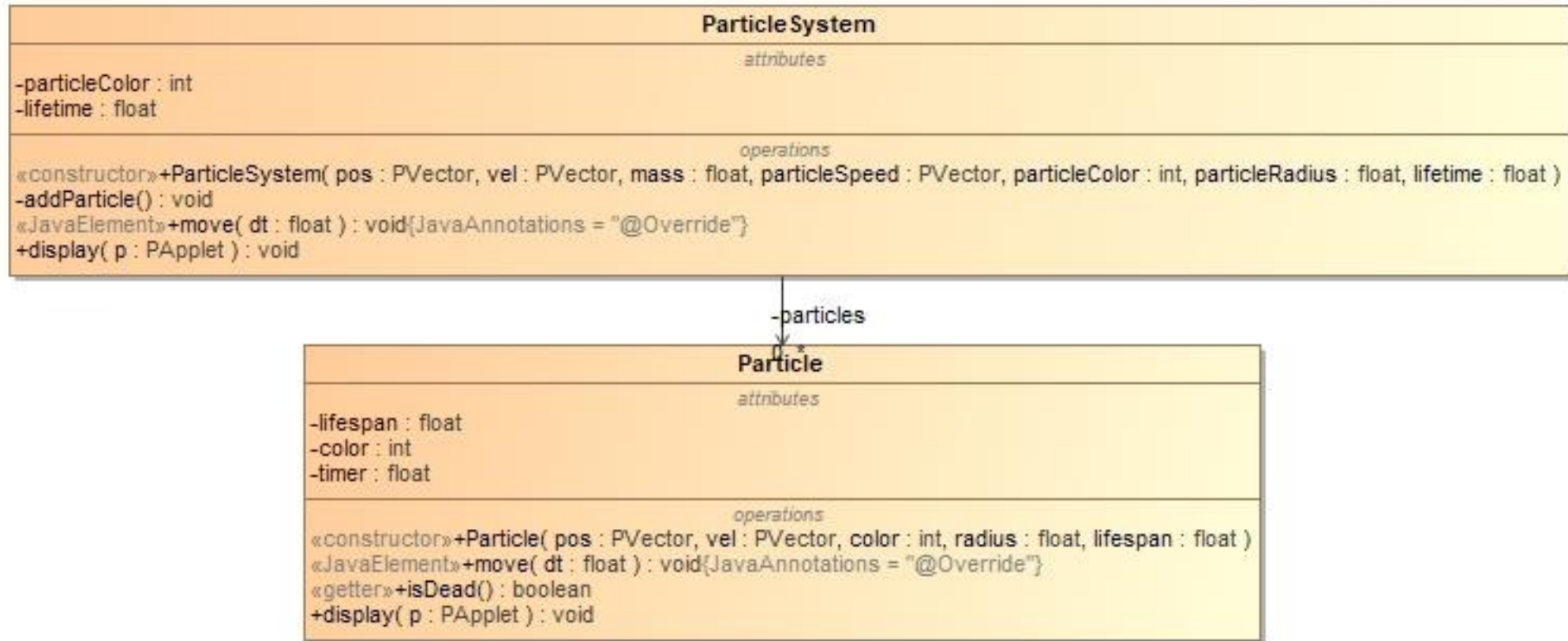


<https://www.lri.fr/~mbi/ENS/IG2/devoir2/files/docs/fuzzyParticles.pdf>

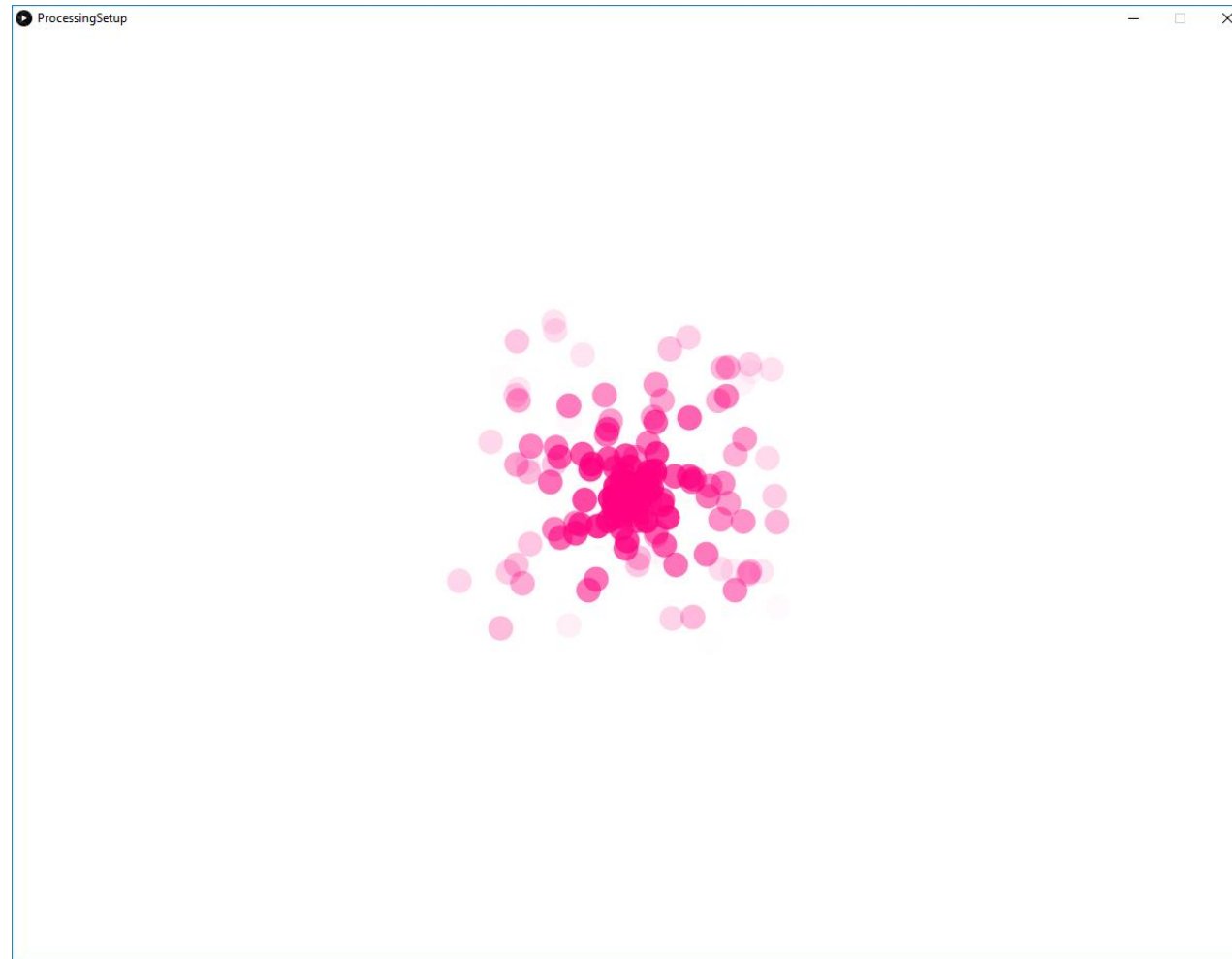
SISTEMA DE PARTÍCULAS

- Criado por William T. Reeves em 1982
 - *Investigador da Lucasfilm*
 - *Star Trek II: The Wrath of Khan*
- Genesis Device
 - Dispositivo com a capacidade de reorganizar matéria e criar um planeta habitável, passível de ser colonizado
 - <https://www.youtube.com/watch?v=52XlyMbxxh8>

SISTEMA DE PARTÍCULAS

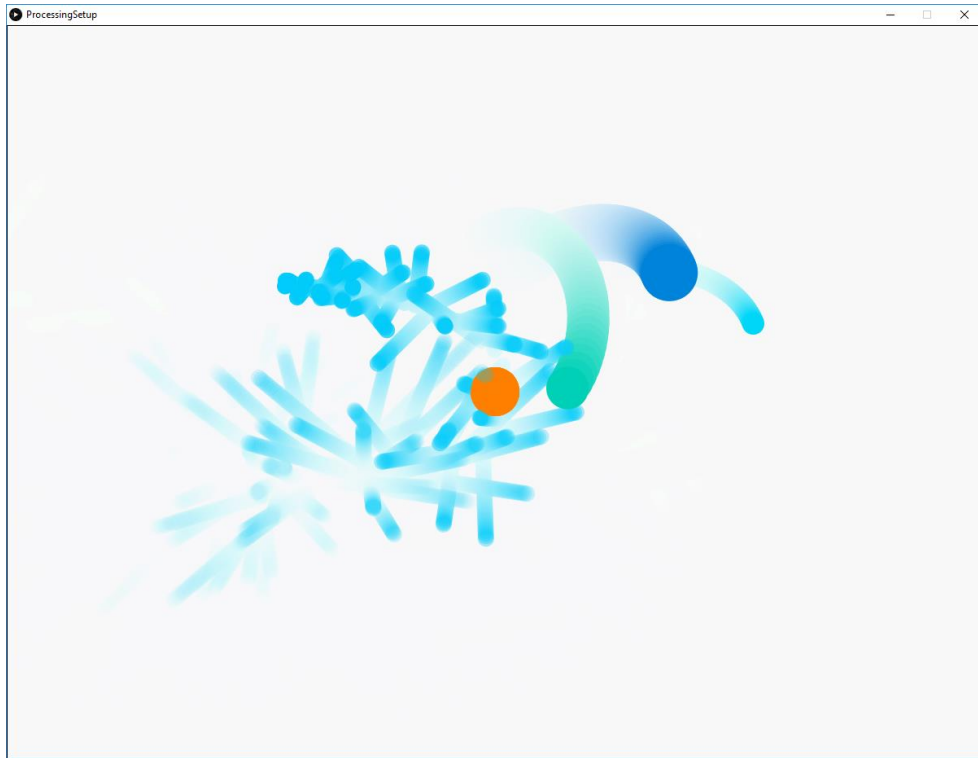


EXERCÍCIO PRÁTICO - PARTICLESYSTEMSAPP



EXERCÍCIO PRÁTICO

- ExplosivePlanetsApp



- FireworksApp

